Technical Workings of Earth Stripes

Nikhilesh Radosevich - nikhilrado.com

Public Codebase: https://github.com/nikhilrado/Earth-Stripes

- Website files contained in /site-files
- File that creates warming stripes /test6.py

Private Codebase: https://github.com/nikhilrado/Earth-Stripes-Business

- Climate Dictionary files in /climate_dictionary
- Earth Stripes Advertising files in /promotion

Public Roadmap: https://sharing.clickup.com/10552948/l/h/a21km-1267/756c96ab4c4577c

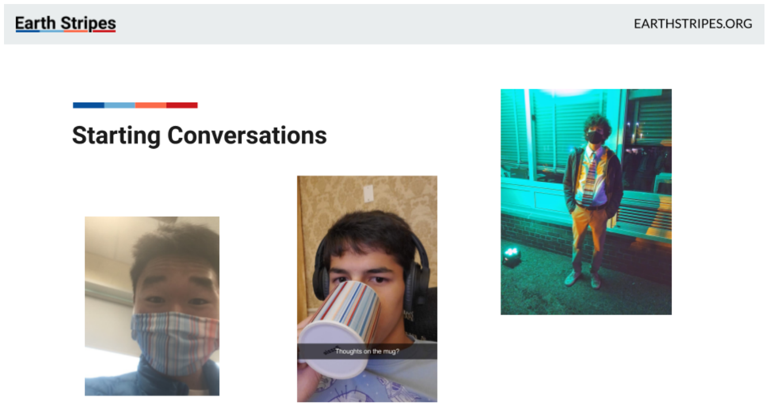
- View all tasks, files, and progress reports

Earth Stripes takes data from multiple sources such as NOAA, the EPA, and Berkeley Earth to create content displayed on the website. The flagship graphics are the "warming stripes" visualizations which were popularized by British climatologist Ed Hawkins.

In this document I'm going to outline eight key processes that keep the website up and running. This is by no means an exhaustive list but hopefully it paints a clearer picture of the inner workings of the website.

Table of Contents

Story



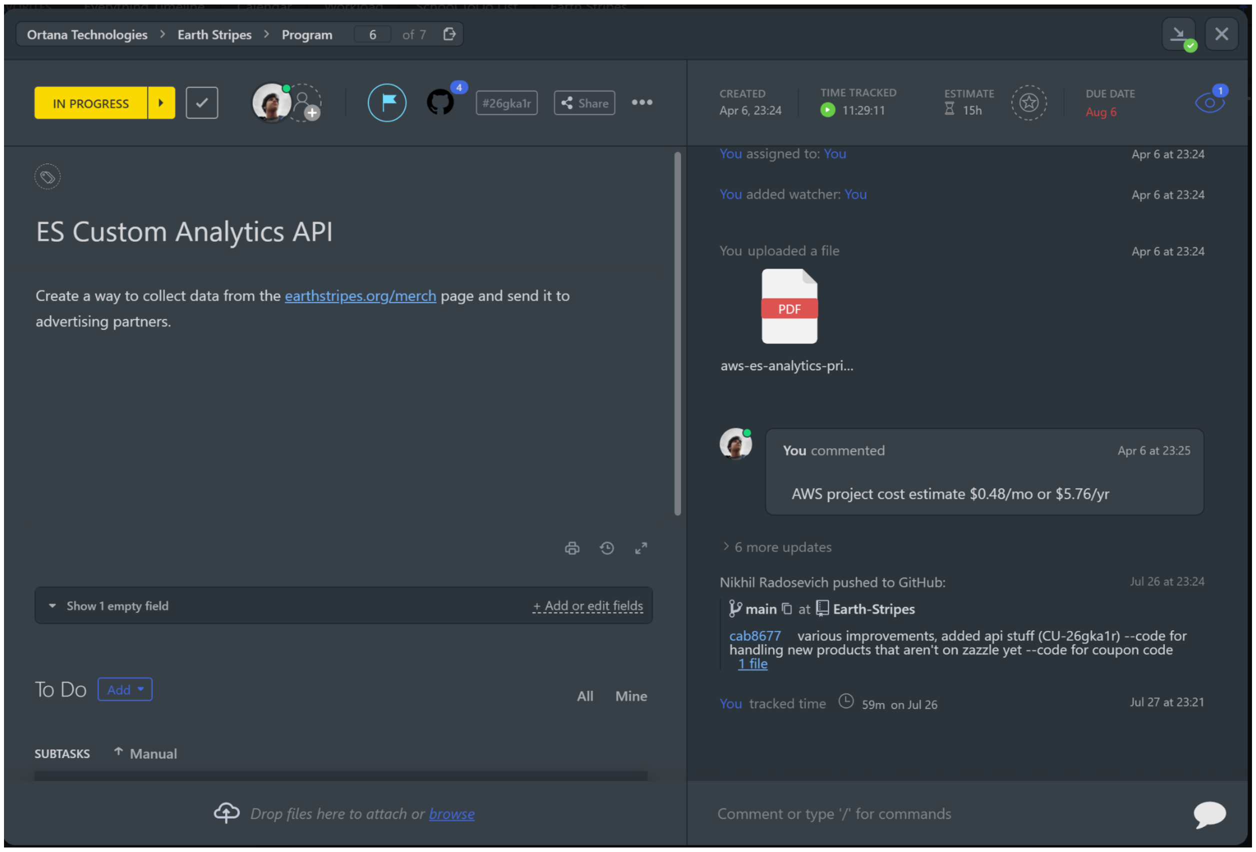
I was scrolling through the news at scout camp when I came across an article about the weather. Since I would be camping in a tent the next 10 days I decided to check it out. At the end of the article the weatherman showed an image of himself wearing a red and blue colored tie, which I learned to be warming stripes.

Throughout that week at camp I was up late at night in my tent scrolling through stack overflow forums trying to see if I could build my own version of the warming stripes. When I got home I raced to my computer to start prototyping. From there, I kept extending the project. First I created a website so anyone could view the stripes then I integrated products, added sharing features, and worked on a bunch of backend systems.

The warming stripes graphics are designed to be a super simple way for the average person to see how the climate is changing (mostly warming) around the world. It isn't just a global issue because the consequences impact people at a local level. Introducing more metrics at lower geographic levels, as well as the stories of climate impacts in those areas can help people recognize the short term effects on the economy, environment, and personal lives. The merchandise provides a fantastic way for people to start conversations about climate change, while helping to fund the website. The idea is that you might be wearing a warming stripes scarf (quite fashionable) and someone might ask about it. Starting climate conversations between people is one of the best ways to educate people. The merchandise sales let me stay independent without relying on donations or financial support from other organizations. I also donate a portion of the revenue to Stripe Climate which funds research into Carbon Capture technologies.

After over 400 hours and 10,000 lines of code across all codebases I'm working on three main goals. First is to expand the data coverage, and depth to include more locations (especially internationally) and visualizations. Second is to work on promotion and education, which involves creating content across platforms that can direct people to the website. Finally, working on more coding and science based efforts will help to extract new insightful data from existing sources. This includes working more with remote sensing data from Google Earth Engine, and geographic details.

Process



ES Custom Analytics API

Create a way to collect data from the earthstripes.org/merch page and send it to advertising partners.

Show 1 empty field + Add or edit fields

To Do Add All Mine

SUBTASKS Manual

Drop files here to attach or browse

You assigned to: You Apr 6 at 23:24

You added watcher: You Apr 6 at 23:24

You uploaded a file Apr 6 at 23:24

PDF aws-es-analytics-pri...

You commented Apr 6 at 23:25

AWS project cost estimate $0.48/mo or $5.76/yr

> 6 more updates

Nikhil Radosevich pushed to GitHub: Jul 26 at 23:24

main at Earth-Stripes

cab8677 various improvements, added api stuff (CU-26gka1r) --code for handling new products that aren't on zazzle yet --code for coupon code

1 file

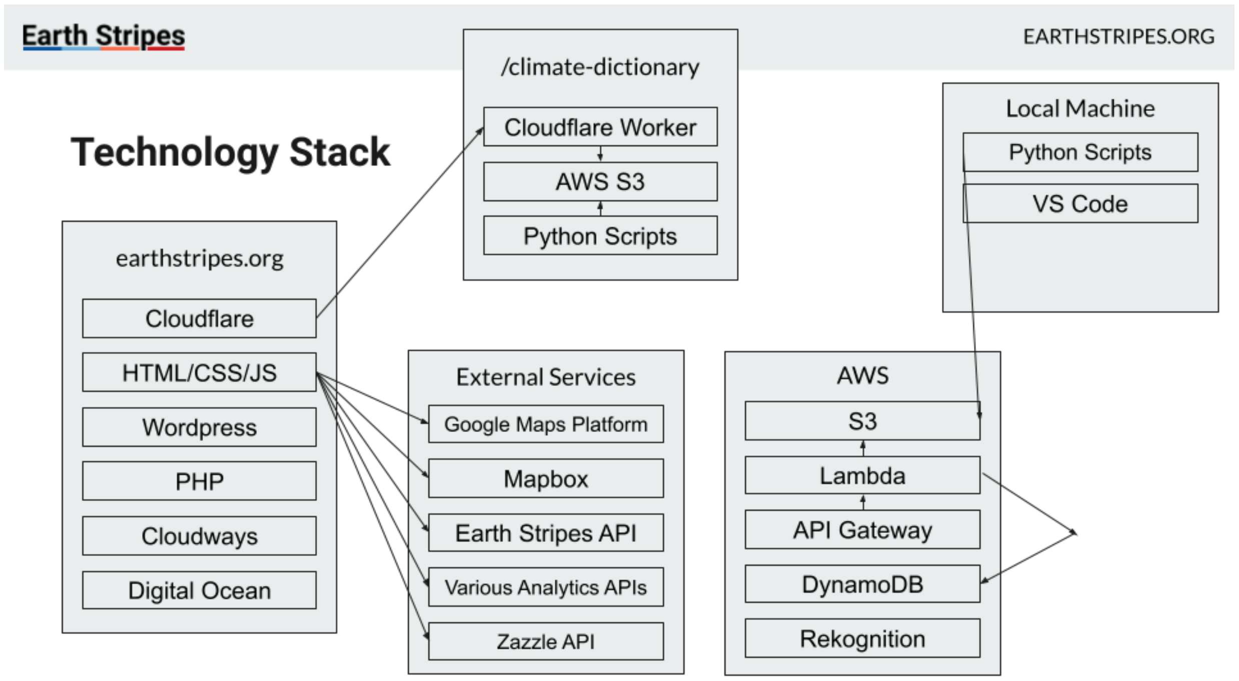
You tracked time 59m on Jul 26 Jul 27 at 23:21

Comment or type '/' for commands

Even though I am the only person currently working on this project, I've tried my best to use proper industry standards when developing, so that when I meet other cool programmers they can easily jump on to help me.

I use GitHub to track all project changes, and ClickUp as my task management software. ClickUp is able to link up to GitHub in order to track project changes. You can view the ClickUp board here, and below is an example of what a task might look like here with comments, documents and GitHub Activity to the right.

Technology Stack



earthstripes.org - main website

- **Cloudflare** - used for DNS, security, CDN, caching, domain management.
- **HTML/CSS/JS** - languages used to construct the website functionality.
- **Wordpress** - content management system used for static content. Also helps with styling of external elements such as headers and footers, as well as making website responsiveness easier.
- **PHP** - server side language used to help modify web pages before server sends them.
- **Cloudways** - hosting as a service provider that provides additional services.
- **Digital Ocean** - underlying server infrastructure.

earthstripes.org/climate-dictionary (in development) - a physically separate website that appears to be hosted on the same domain as the main website.

- **Cloudflare Workers** - intercepts traffic to earthstripes.org at the edge, and requests content from the s3 bucket that hosts the content for the Climate Dictionary.
- **AWS S3** - hosts the static html/css/js files that are generated on my computer.

External Services

- **Google Maps Platform** - provides autocomplete and GeoData on the /merch page.
- **Mapbox** - provides autocomplete and GeoData on the home page.
- **Earth Stripes API** - collects analytics and usage data to be stored.
- **Various Analytics APIs** - Google Analytics, Hotjar, Meta Pixel, Pinterest Pixel, Twitter conversion tracking.
- **Zazzle API** - used to dynamically create products.

Amazon Web Services

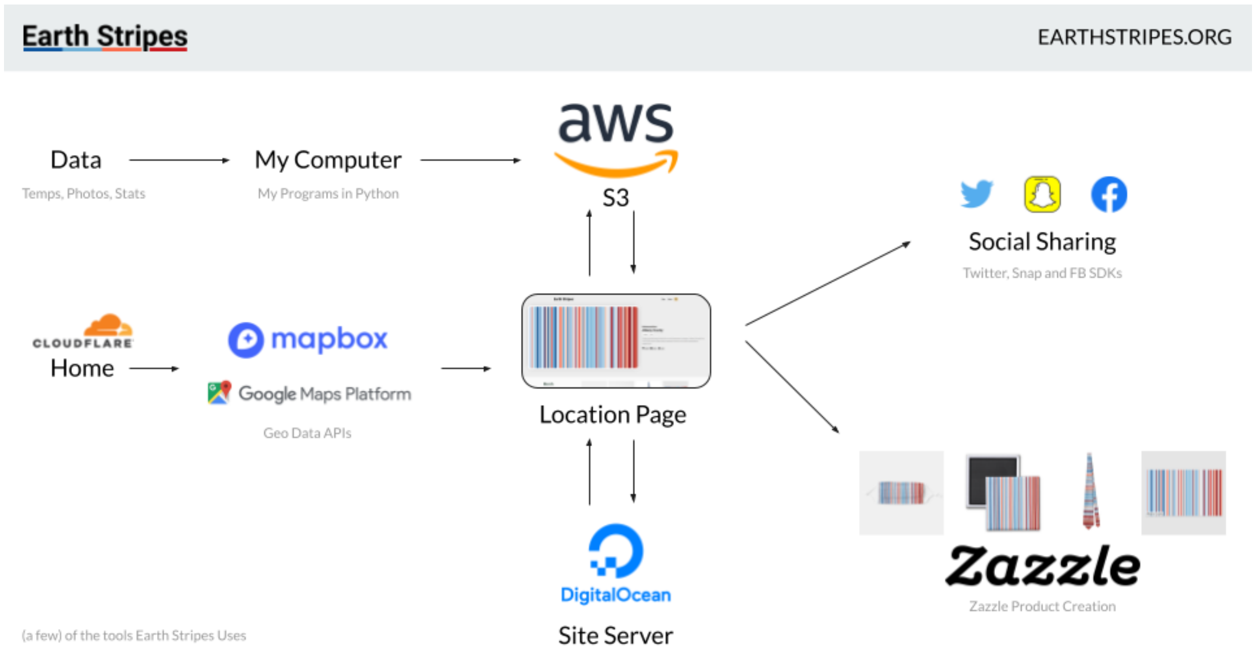
- **Super Simple Storage (S3)** - Hosts all of the data for the website, over 100k files in total. This basically acts as a MongoDB as data is stored in json files, however I didn't know what MongoDB was when I created the website.

- **Lambda** - hosts serverless functions such as the Earth Stripes Analytics service, price collection and advertising service, and the twitter share service.
- **API Gateway** - sits on top of lambda functions to add domain, rate limiting + security
- **DynamoDB** - fast and dynamic database used for storing data from the Earth Stripes Analytics service.
- **Rekognition** - there are many images of people's faces in our libraries. Cropping their faces perfectly would take a lot of work, so we use AWS' ML Computer Vision engine to detect faces and crop them properly.

Local Machine - runs the python scripts, math, generates warming stripes, aggregates data

- Python Libraries
    - os, json, csv, requests, math, base64, datetime - utility modules
    - Python Pillow - Image manipulation. Mainly used to create warming stripes and modify/crop graphics for the website.
    - Boto3 - Amazon's Python SDK
    - ssh - server access
    - pandas - data analysis, graphing and modeling
    - numpy - math calculations

Retrieving Location Pages (/Site Files/results-page)



Problem:
There are hundreds of thousands of locations on the Earth, all spelled in very different ways. I need to find a way to route people to the correct page that is both simple and fast.

Solution:
When someone first arrives at the home page, they are directed to Cloudflare which serves as the CDN and security service that helps cache and protect the site from attacks. Users are greeted with a simple search bar interface, modeled after Google.

Users can either use the “Your Location” button which uses the browser Geolocation API to get coordinates or begin typing in the search box. After each keystroke a request (L:224) is sent to MapBox which returns autocomplete results. After a location is selected we will request the location hierarchy from Mapbox providing the place id (from search) or the longitude and latitude coordinates (from geolocate).

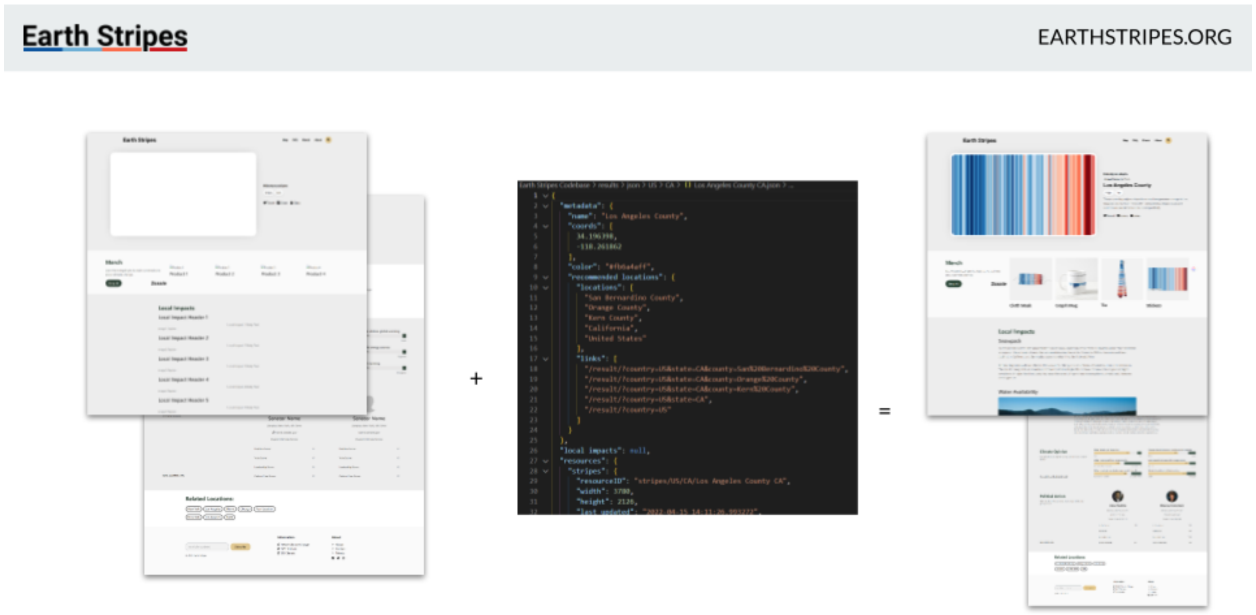
The location hierarchy looks something like this: “country>state/province>city>census tract, congressional district.” From there we use an algorithm which finds the lowest geographical location that Earth Stripes has data on. For example entering Nairobi, Kenya will redirect you to a page for Kenya because we don’t have data for cities in Kenya, while a search for “Albany, New York” will return a page for Albany County, NY.

```
https://www.earthstripes.org/result/?country=US&state=MA&county=Middlesex%20County
```

There are hundreds of thousands of locations to search from, so each result page is the same HTML file. The location is specified using URL query parameters so that a) user history is preserved, b) search engines can index pages, and c) social media sharing results in direction to the same page.

Once the HTML/CSS/JS have been loaded the page makes an AJAX request to a JSON file hosted on AWS S3 containing the specific location information. This file is formatted in such a way that it can be accessed directly, without an API as shown below.

```
https://earthstripes.s3.us-east-2.amazonaws.com/v3/json/US/MA/Middlesex+County+MA.json
```



Earth Stripes Codebase > results > json > US > CA > Los Angeles County CA.json > ...

```
{
  "metadata": {
    "name": "Los Angeles County",
    "coords": [
      34.196398,
      -118.261862
    ],
    "color": "#fb6a4aff",
    "recommended locations": {
      "locations": [
        "San Bernardino County",
        "Orange County",
        "Kern County",
        "California",
        "United States"
      ],
      "links": [
        "/result/?country=US&state=CA&county=San%20Bernardino%20County",
        "/result/?country=US&state=CA&county=Orange%20County",
        "/result/?country=US&state=CA&county=Kern%20County",
        "/result/?country=US&state=CA",
        "/result/?country=US"
      ]
    }
  },
  "local impacts": null,
  "resources": {
    "stripes": {
      "resourceID": "stripes/US/CA/Los Angeles County CA",
      "width": 3780,
      "height": 2126,
      "last updated": "2022-04-15 14:11:26.993272",
```

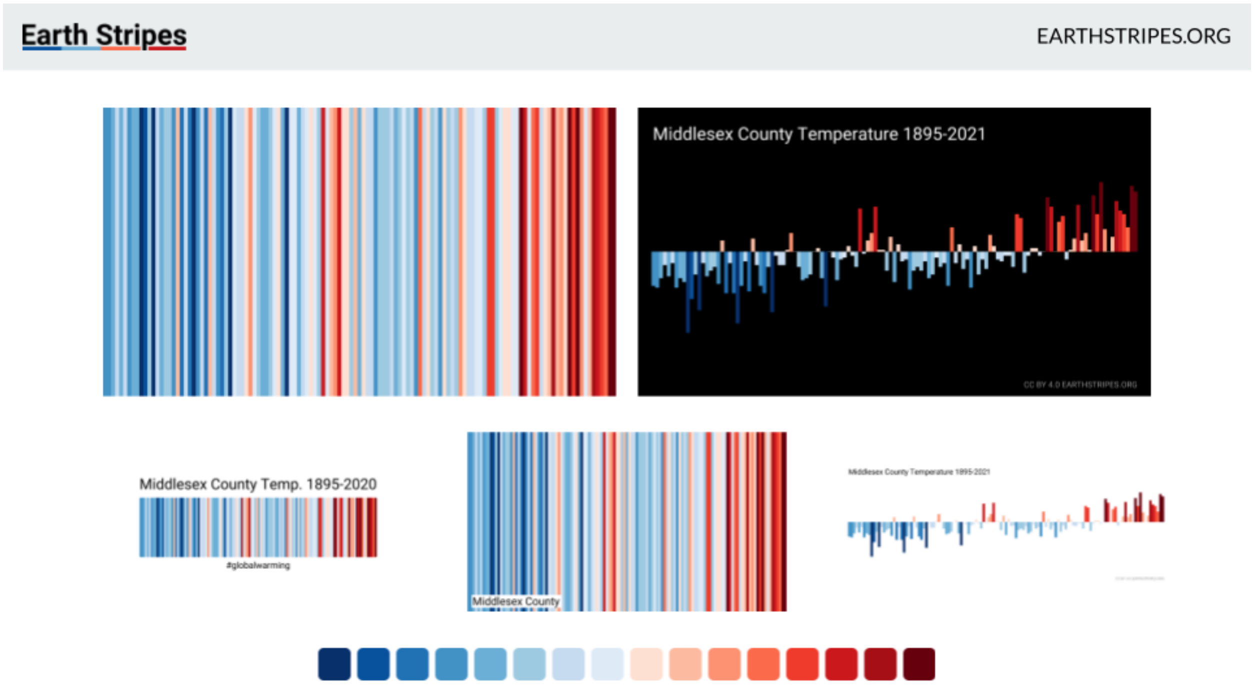
Blank page JSON Document Final Location Webpage

After the location JSON file is downloaded the JS will parse it and fill in (or hide) areas of the page where data is present. Some places don't have energy data, so those sections of the page will be deleted.

Product Creation

Products are created with the Zazzle API. When a user arrives at a location page, a link to a product will be created by appending URL Query Parameters to a zazzle template. Multiple versions of each stripe variant exist as images in order to fit different product sizes. Users simply click on the link generated and are able to purchase merchandise through Zazzle.

Warming Stripe Generation (test6.py, test30_generateImages.py, createSVG.py)

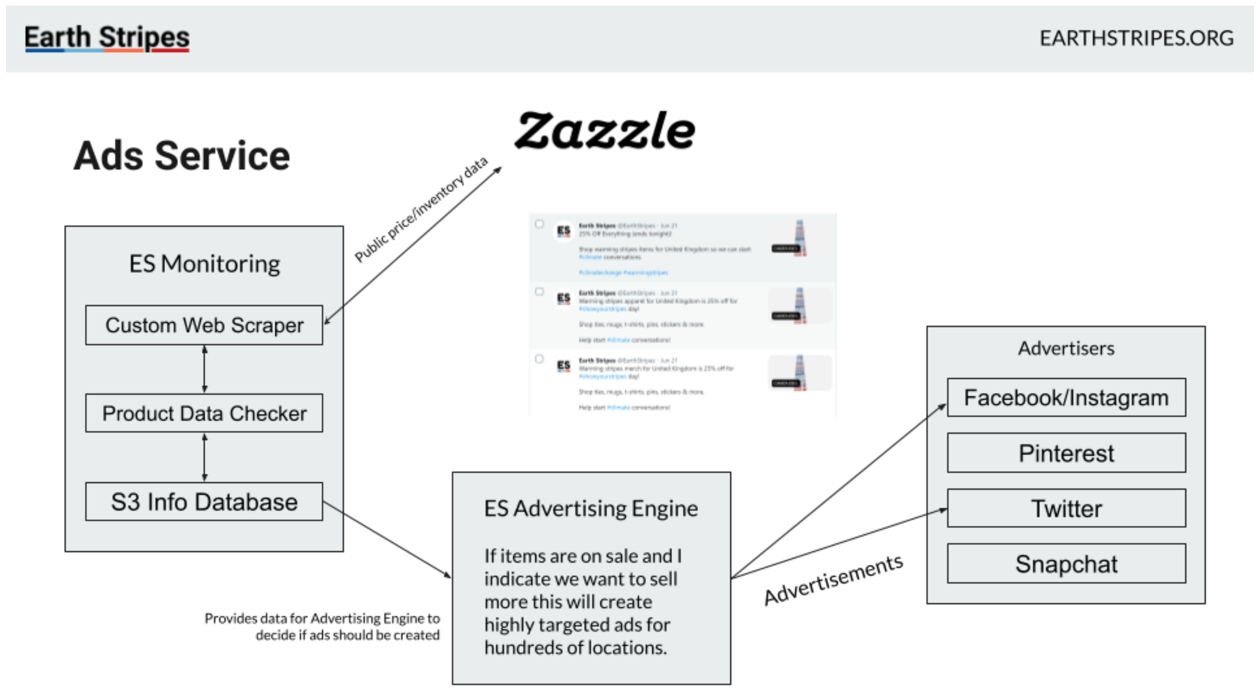


Temperature data from different sources is categorized, normalized and then sorted into 16 different color buckets. The color bars are relative meaning they just show the temperature change in each area, and should not be used to compare areas.

The mean temperature from 1901-2000 is taken as the baseline reference period. The max (dark red) and min (dark blue) colors are 2.6 standard deviations from the baseline. From there on anything above the mean is divided into 8 different color buckets and graphed. The same is done for the temperatures below the mean. After these values are determined the data is inputted into a graphic specific function in order to make the graphics.

There are 12 different graphic types in different sizes and formats. The first two pictured above are displayed on the website. The first of three is used when someone shares a location on Snapchat, the middle one is used when images are shared on Twitter, and the last one is used on the mugs that are sold.

Earth Stripes Advertising API (code not public)

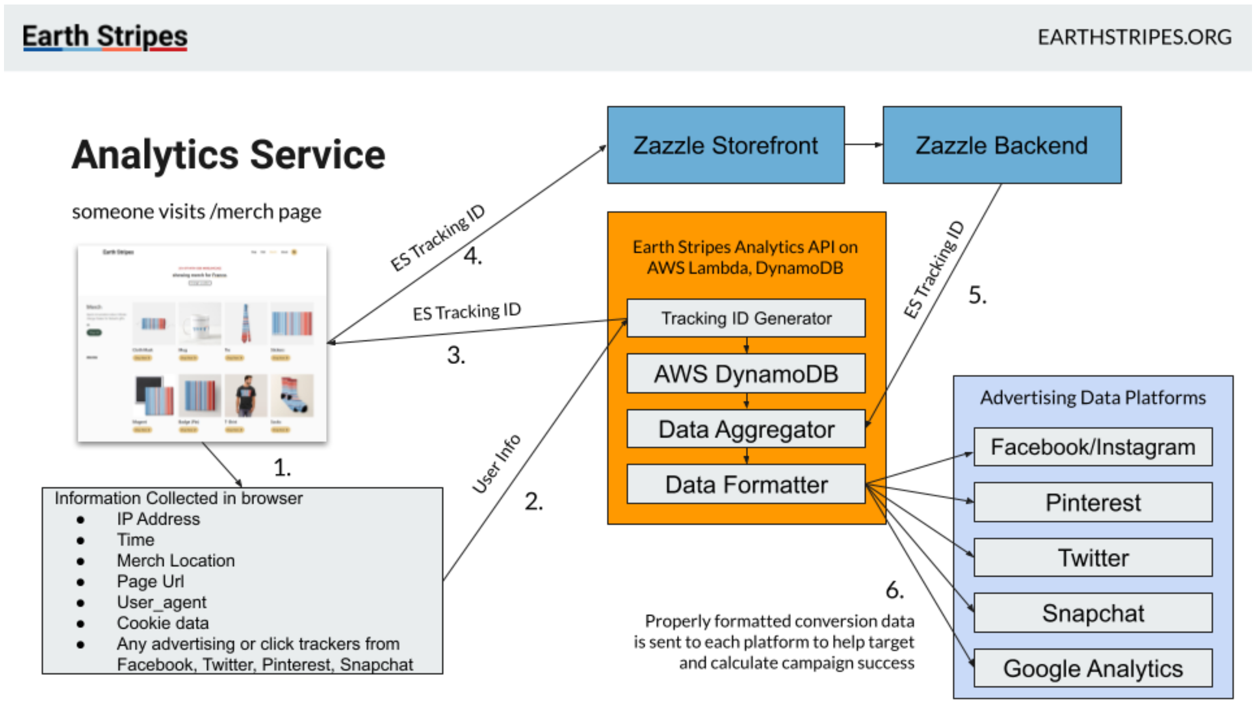

Earth Stripes
EARTHSTRIPES.ORG
Ads Service
Zazzle
Public price/inventory data
ES Monitoring
Custom Web Scraper
Product Data Checker
S3 Info Database
Provides data for Advertising Engine to decide if ads should be created
ES Advertising Engine
If items are on sale and I indicate we want to sell more this will create highly targeted ads for hundreds of locations.
Advertisements
Advertisers
Facebook/Instagram
Pinterest
Twitter
Snapchat

Problem:
There are hundreds of products available for purchase across many different countries, and localized advertisements will help people relate more to the product.

Solution:
In order to advertise things in an effective manner, scraper bots that I've created go to the Zazzle website each day and scrape product data. When items go on sale or prices change, it signals my script to start updating our product catalog. The script then sends the data to another script which generates hundreds of personalized products and targets them to the people living in their respective locations.

Result:
The technology works fine, however it appears that there isn't enough data to properly target ads, and I currently don't have enough money to test out different methods. These files were shelved and will be used in the future.

Earth Stripes Analytics Service (code not public)


Earth Stripes
EARTHSTRIPES.ORG
Analytics Service
someone visits /merch page
1.
Information Collected in browser
IP Address
Time
Merch Location
Page Url
User_agent
Cookie data
Any advertising or click trackers from Facebook, Twitter, Pinterest, Snapchat
2.
User Info
3.
ES Tracking ID
4.
ES Tracking ID
Zazzle Storefront
Zazzle Backend
5.
ES Tracking ID
Earth Stripes Analytics API on AWS Lambda, DynamoDB
Tracking ID Generator
AWS DynamoDB
Data Aggregator
Data Formatter
6.
Properly formatted conversion data is sent to each platform to help target and calculate campaign success
Advertising Data Platforms
Facebook/Instagram
Pinterest
Twitter
Snapchat
Google Analytics

Problem:
Many ecommerce stores (such as those on shopify or woocommerce) have integrations to help collect data. Since Earth Stripes partners with Zazzle to offer products, it is very difficult to see which users are buying your products. This makes it difficult to optimize for maximum Revenue.

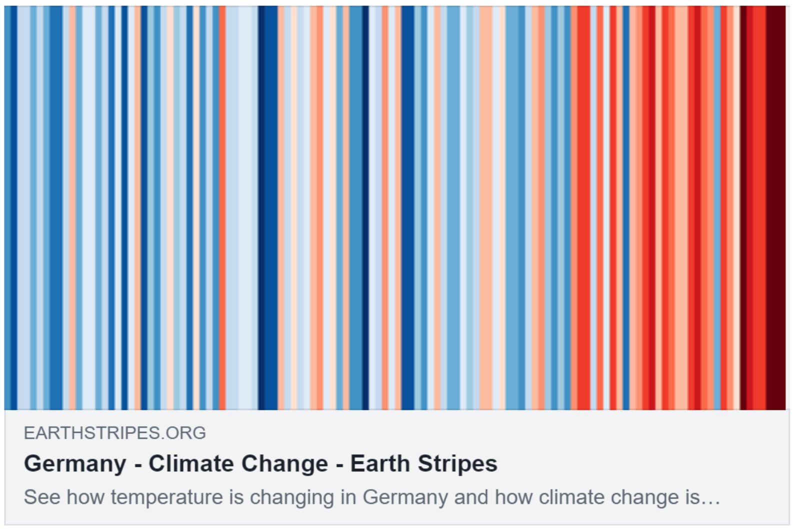
Solution:
A custom built Earth Stripes Analytics Service that collects, stores, processes, and analyzes data before formatting it and sending it off to advertising platforms to track conversion rates.

When a user visits the merch page a request is made to the Earth Stripes Analytics API with identifiable browser information (IP address, time, user agent, cookie data, analytics tracker ids) and purchase information (url, what location are they looking at for the stripes). The ES Analytics API returns an Earth Stripes Tracking ID which is a mostly unique uuid, while the data along with the uuid is stored in a DynamoDB. When a user purchases an item the uuid is passed to Zazzle who then records it if a purchase is made.

After a purchase is made sales data is downloaded from Zazzle and the data from DynamoDB is then matched with the purchase data to create a master purchase record file. From there a data formatter reformats the data for each advertising data platform and sends the data. Now we can measure the ROI of each advertising campaign.

Result:
The technology works fine, and will allow us to collect more data until we can spend more money on advertisements.

Social Sharing (functions.php)


EARTHSTRIPES.ORG

Germany - Climate Change - Earth Stripes

See how temperature is changing in Germany and how climate change is…

Problem:
When I created the website I always wanted it to be incredibly easy for people to share their findings on social media. Usually when someone shares something on social media the platform (say Twitter) will access open graph tags in the html head which contains information such as the title, cover image, and link.

Since all content on the earthstripes.org/result page is the same until JavaScript alters it each person would see a generic page.

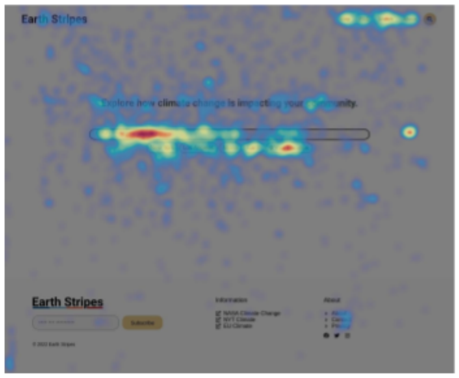
Solution:
Media crawlers don’t execute JavaScript (Google’s crawlers do) so I needed to find a way to change the HTML of the page before it gets served to the user. Thankfully the location data is coded into the result page url via query parameters. I was able to write a function in php, that will intercept requests to the /result page and rewrite the open graph metadata.

I also generated custom graphics for social media platforms such as Twitter so that the warming stripes for each page would be specific to that location. The Image above displays the warming stripes for Germany, not just generic ones when shared on Facebook.

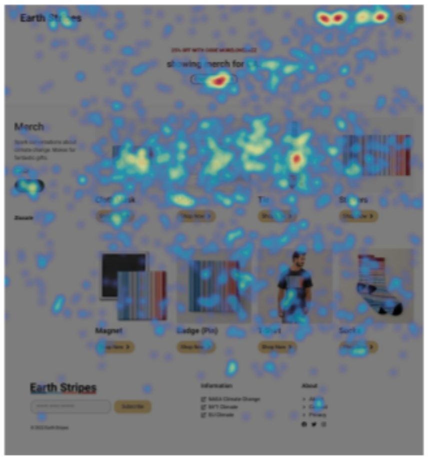
Accessibility & User Experience

Earth Stripes EARTHSTRIPES.ORG

User Experience



Home Page



Merch Page

Result Page

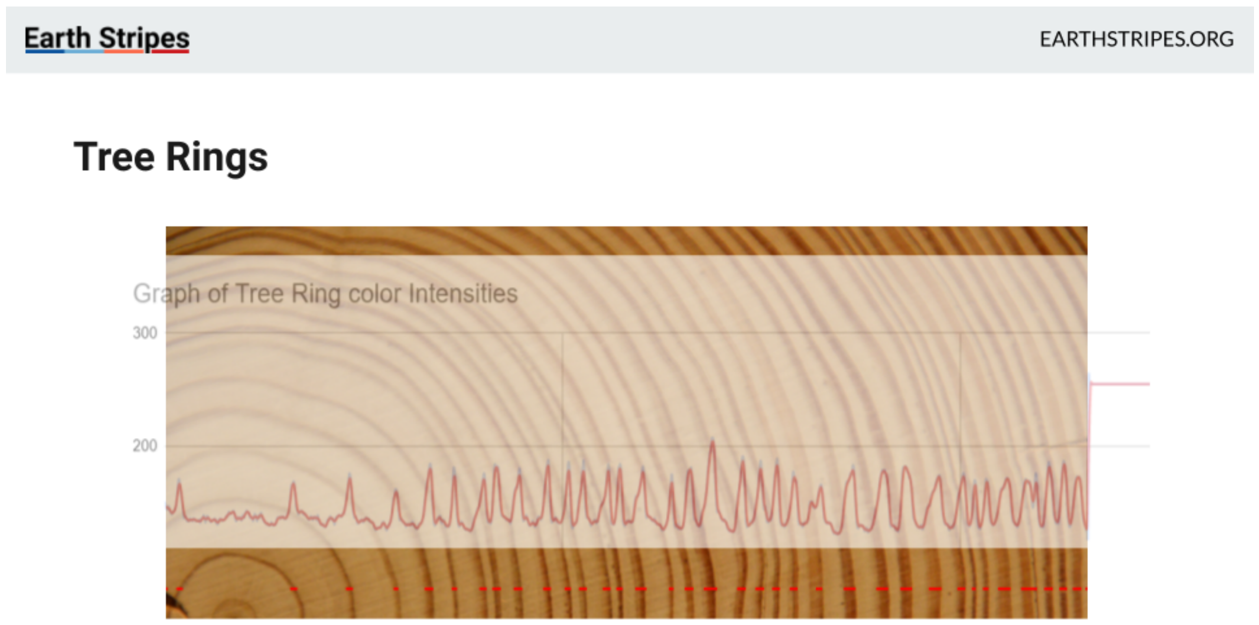
Making content accessible to people, bots, and screen readers is incredibly important. Using external tools such as Hotjar helps me see how my users use the website. Some people may have a hard time navigating to certain pages, others may not find information useful. Taking a look at how others use your website can really help improve user experience.

A user shouldn't have to think too hard about what is going to happen. Using responsive design with clear elements and action buttons keeps visitors happy. As I've been programming this project I've periodically looked at how users are using it so that I can tweak the design in ways that make everyone happy. Everything from the color of the buttons to the radius of the search bar has slowly evolved as users come and go.

I'm also happy to say that I work really hard to make the site accessible. It can be difficult to do for websites with dynamic content like this, but in the end it pays off.

In the future I'd like to work towards making the website accessible in more languages. Only a small portion of people in the world speak English as their primary language, so opening the website up to more people is always fantastic, plus people in Europe tend to purchase a lot of warming stripes merchandise.

Tree Rings



Today almost all of the temperature and humidity measurements are incredibly accurate as they are taken by digital thermometers and or satellites. However climate is the weather conditions over a long period of time, so gathering historical data is incredibly important.

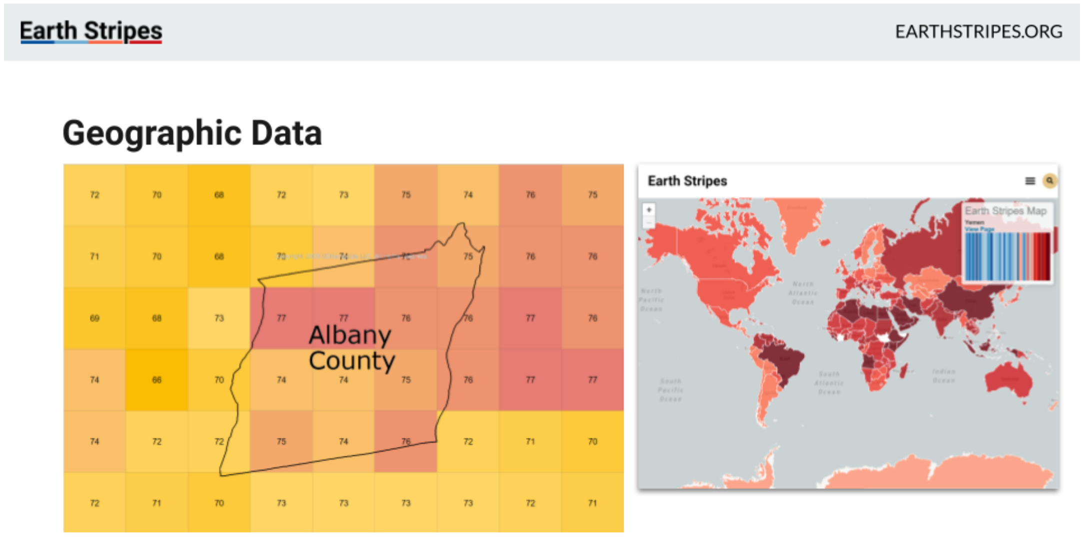
Scientists use numerous different proxies to help them estimate the past temperature. One of these techniques involves the study of tree rings known as dendrochronology. Looking at the color, spacing, width, number and shape of rings can give people a lot of information.

I am trying to make an API that will allow users to take a picture of a tree stump, upload it and receive information about the tree. A very simple machine learning model is used to identify the bounds of the tree stump in the photo, and will also identify a quarter. The quarter allows us to get an incredibly accurate object to calculate the size of the tree as quarters are incredibly common and durable.

From there we can measure the tree rings by graphing the color intensities as shown in the image above. From there we can do some relatively simple math from calculus that will allow us to figure out where each tree ring starts and ends allowing us to extract additional information.

Ideally once this works I will make an app or website where any user can help record data. From there on maybe I can start to create other tools to help scientists if I continue to advance my understanding of climate measurements.

Geographic Data & New Insights (/data-connectors, manageJSON.py, /map-stuff)



In order to import new data into Earth Stripes, I create a new “data connector” file which takes data from an external source, downloads it in its raw form, then begins to process it. Data will be trimmed, reformatted, translated, and averaged before it is passed onto another file manageJSON.py which helps act as the content management system. It is responsible for organizing, keeping track of, and attributing data to the correct source before sending each piece of data to the location document where it is stored to be displayed on our website and other surfaces.

Other data requires some additional processing. Many sources of climate information are housed in massive NetCDF files which can be organized in many different ways. They store geographical data across different time scales in a grid format similar to the image above. In order to translate these data points into human understandable metrics, we need to take the NetCDF data of millions of data points organized in geometric grids and process them.

In the example above we see Albany County, NY superimposed above a fictional average temperature grid. The borders for Albany County can be extracted from the Open Street Maps API as a shapefile, which is basically just a list of coordinates in a series that draw a polygon.

When each border can be tens of thousands of points long I needed to find a way to get a weighted average of the cells inside the area. In order to do this I first simplified the high-resolution shapefiles by removing ~90% of the border depending on the distance between points. So if the distance between two points was large, both points would be kept, if it was relatively small I could remove some of the points. From there I used some Riemann sum like area calculations that I learned from calculus class to estimate what percent of the shape covered each cell and obtained an average of the data to assign to the location. From there it is passed on to ManageJSON.py to be added to the location JSON file.